FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Performance of side-channel attacks on encrypted 802.11 web traffic and impact of network-level noise

Pedro Fonseca

WORKING VERSION

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Ricardo Morla

July 24, 2017

Resumo

No mundo de hoje, as redes WiFi são comuns e estão ligadas ao nosso dia-a-dia: acedemos-lhes com os nossos *smartphones*, os nossos *tablets* e os nossos portáteis. Isto significa que inevitavelmente muitos dados sensíveis são transmitidos nestas redes. Um ataque malicioso convencional tentaria aceder a estes dados. Um exemplo seria o de um *hacker* a tentar desencriptar as comunicações de um utilizador para tentar obter a sua password.

A encriptação, que está presente em virtualmente todas as redes WiFi, pode ajudar a proteger os dados de um utilizador de acessos externos. No entanto, através de ataques *side-channel* é possível obter informação e quebrar a privacidade de uma pessoa, independentemente da encriptação. Estes ataques diferem dos seus equivalentes convencionais no sentido de que o atacante está fora do sistema e não tenta obter acesso a ele. De facto, um ataque *side-channel* baseia-se nas características dos dados transferidos, que podem ser observados desde fora, em vez dos conteúdos.

Esta tese tem o propósito de planear um destes ataques. O objetivo do ataque é identificar o *website* a que um dado utilizador está a aceder, tendo acesso apenas ao tráfego WiFi encriptado que o dispositivo do utilizador está a gerar. O tráfego pode ser capturado e processado por um computador com uma placa de rede com capacidades WiFi, que precisa de estar nas proximidades do dispositivo alvo. Em paralelo ao acesso ao website ruído ao nível da rede, como *streaming* de música ou DHCP, pode existir. Por outro lado, as características do *website* - que pode ser mais ou menos estático - pode torná-lo mais difícil de identificar. O grau de influência que estes factores poderão ter na identificação de *websites* é estudada nesta tese.

Websites podem ser caraterizados pelas suas capturas de tráfego. A partir destas capturas pode-se calcular um conjunto de *features*, como o número de *bytes* ou os pacotes por segundo. De forma a conseguir identificar os *websites*, o primeiro passo é constuir um *dataset* com as *features* das capturas realizadas anteriormente para cada um dos *websites* selecionados para o ataque. De seguida, um algoritmo de *Machine Learning* que seja capaz de de prever o *website* a que uma nova captura pertence com base em dados anterior é desenvolvido e aplicado.

De forma a realizar testes, um *dataset* foi construído com as *features* de capturas pertencentes a dez *websites* diferentes, sendo que metade são estáticos e a outra metade são mais dinâmicos. Cinco níveis de ruído foram considerados, incluindo o caso em que não existe ruído.

Testes efetuados com as capturas mostram que um atacante pode identificar qualquer um dos *websites* utilizando o algoritmo desenvolvido para esta tese. O *score* F1, que é uma métrica para medir a eficácia dos testes, é 77% quando o ruído não é considerado e 69% quando é.

Os resultados mostram que as instâncias em que o algoritmo toma um *website* por outro ocorrem principalmente dentro da mesma classe - estático ou dinâmico. Isto significa que *websites* dinâmicos raramente são confundidos com estáticos, e vice-versa.

Quando o ruído é considerado, os resultados não são conclusivos. A deteção de alguns *websites* permanece relativamente inafetada pelo ruído, enquanto que outros mostram um aumento ou diminuição da eficácia em certos níveis de ruído. Não existe um padrão uniforme, o que significa que mais testes com diferentes tipos de ruídos são necessários.

Abstract

In today's world, WiFi networks are widespread and intertwined with our daily lives: we access them through our smartphones, our tablets and our laptops. This means that inevitably a lot of sensitive data is transferred through these networks. A conventional malicious attack would attempt to gain access to such data. An example would be that of an hacker trying to decrypt a user's communications in order to obtain his password.

Encryption, which is present in virtually every Wi-Fi network, can help protect a user's data from outside access. Nonetheless, through side-channel attacks it is possible to obtain information and potentially breach a user's privacy, regardless of the encryption. These attacks differ from their more conventional counterparts in that the attacker is outside the system and does not attempt to gain access to it. In fact, a side-channel attack relies on the characteristics of the transferred data, which can be observed from the outside, instead of its content.

This thesis has the purpose of devising such an attack. The objective of the attack is to identify the website a given user is accessing, having access only to the encrypted WiFi traffic the user's device is generating. The traffic can be captured and processed by a computer with a WiFi-capable network card, which needs to be near the target's device. In parallel with the accessing of the website, network-level noise, such as music streaming or DHCP traffic, may exist. On the other hand, the characteristics of the website - it can be more or less static - can make it more difficult to identify. The degree of influence these factors may have on website identification is studied in this thesis.

Websites can be characterized by their traffic captures. From these captures one can calculate a set of features, such as bytecount and the average packet size. In order to be able to identify websites, the first step is to build a dataset with the features of captures performed beforehand for each of the websites selected for the attack. Afterwards, a machine learning algorithm that can predict the website a new capture belongs to based on the previous data is applied.

In order to perform testing, a dataset was built with the features of the captures belonging to ten different websites, with half being static and half being more dynamic. Five levels of noise were considered, including the case in which there is no noise.

Testing performed with the captures shows that an attacker can identify any of the websites using the algorithm developed in this thesis. The average F1 score, which is a metric to measure the accuracy of the tests, is 77% when noise isn't considered and 69% when it is.

The results show that the instances in which the algorithm mistakes a website for another mostly occur within the same class - static or dynamic. This means that dynamic websites are rarely confused with static websites, and vice-versa.

When noise is considered, the results are not conclusive. The detection of some websites remains relatively unaffected by the noise, while others show an increase or a decrease of the accuracy on certain noise levels. There is no uniform pattern across the websites, which means that further testing with different types of noise is needed.

Agradecimentos

Quero agradecer aos meus amigos e colegas da FEUP pelo convívio, pela ajuda e pelos bons tempos que vivemos juntos. Uma nota especial para o pessoal da I224, essa lendária sala onde muito se trabalhou (entre outras coisas).

Quero também agradecer aos meus amigos de longa data por estarem presentes desde sempre, nos bons e nos maus momentos.

Ao Professor Ricardo Morla, que sempre me ajudou e guiou durante este projeto, deixo aqui um agradecimento especial.

O agradecimento final e mais sentido deixo para a minha família, pelo apoio incondicional que sempre me deram.

Pedro Fonseca

"Why is it that when one man builds a wall, the next man immediately needs to know what's on the other side?"

George R.R. Martin

Contents

List of Figures

List of Tables

Abbreviations

3G	Third Generation of the Mobile Communications Protocol
ARQ	Automatic Repeat Request
AP	Access Point
CSI	Channel State Information
DHCP	Dynamic Host Configuration Protocol
DTW	Dynamic Time Warping
FCUP	Faculty of Science of the University of Porto
FDUP	Faculty of Law of the University of Porto
FLUP	Faculty of Arts of the University of Porto
FEUP	Faculty of Engineering of the University of Porto
FMUP	Faculty of Medicine of the University of Porto
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
LAN	Local Area Network
MAC	Media Access Control
ML	Machine Learning
OSI	Open Systems Interconnection
PDML	Packet Description Markup Language
PHY	Physical Layer
SSH	Secure Shell
SSL	Secure Socket Layer
SCTP	Stream Control Transmission Protocol
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunication System
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access
XML	Extensible Markup Language

Chapter 1

Introduction

1.1 Context

WiFi, also known as 802.11, is the *de facto* standard for wireless communications, as it is nowadays widely used by many people, especially in developed countries. WiFi networks can be found anywhere, from regular households to public places such as airports or libraries, and they are easily accessible to most people. In fact, 64% of the population in the United States owns a smartphone [4], which is one of many devices capable of connecting to these networks. A natural consequence of these facts is that a large volume of Internet traffic flows through WiFi networks (as much as 68% of all traffic by 2017 [5]), which means that a lot of sensitive data may be transmitted through them. Activities such as shopping and social networking entail the transmission of private information that could be used by an attacker.

Internet users are getting more and more worried about their privacy, with as many as 86% of them having taken steps to remove or mask their Internet footprints [6]. However, there is next to no awareness of the dangers and possible information leaks that can come from the usage of a WiFi network.

WiFi networks are considered secure, with the vast majority of them offering encryption such as WPA or WPA2. One may think that this means they are safe, but in fact they are not. Even without breaking the encryption, there are ways to extract potentially dangerous information simply by passively collecting data from the "outside". This type of attack is called "side-channel", and allows an attacker who is eavesdropping on a WiFi network to obtain compromising information of an unsuspecting person, which may then be used for blackmailing, extortion or phishing scams.

Current 802.11 standards do not contemplate any counter-measures to this type of attack, which are only recently being studied. This means that more likely than not, anyone who uses a WiFi-enabled device such as a smartphone or a laptop is possibly exposed and may be the target of a side-channel attack.

1.2 Motivation

In a world in which more and more of our communications are done through WiFi networks, it is of vital importance to ensure that all network traffic carrying sensitive information is as secure and private as possible. Side-channel attacks are a big security threat since they allow profiling and activity monitoring of targets who are using a wireless network, while they remain unaware of the privacy breach. Furthermore, encrypting the traffic does not stop side-channel information leaks from occurring, since they are based in characteristics that remain largely independent from the type of encryption.

This work aims to raise awareness about the vulnerability of WiFi networks to this type of attack, so that effective counter-measures may be implemented.

1.3 Objectives and contributions

The final objective of this thesis is to develop a solution which can detect and classify WiFi traffic, through the perspective of an outside attacker. This solution, based on machine learning, allows for specific characterization of an individual's activity in terms of which websites he is accessing.

Websites can be more or less static, depending on the content they have. In this thesis the testing is performed for both dynamic and mostly static websites, so as to determine if the accuracy of the classification depends on this characteristic.

The influence of network-level noise on the accuracy of the classification of WiFi traffic is also studied in this thesis. Several levels of noise are considered, so as to determine how the accuracy changes with the increase of the noise.

1.4 Dissertation Structure

On Chapter 2 a general review of the theoretical elements that are essential to this thesis are explained, including important concepts such as the functioning of WiFi and the structure of a side-channel attacks on 802.11. Examples of implementations of side-channel attacks are also described. Chapter **??** explains the problem which this thesis seeks to address. An example of a real world scenario of an attack is presented. On the chapter 3 the proposed solution is presented in mostly theoretical terms. Chapter 4 pertains to the methodology and experimental procedures regarding the capture of traffic and its subsequent processing. The obtained results are analyzed on chapter 5, with the aid of visual representations such as graphs and matrices. Finally, the conclusions and analysis of the results are made on 6.

Chapter 2

Literature Review

There are a number of concepts that one needs to understand before reading about the implementation of this thesis, and as such in the following sections they will be presented and explained. Some are more general than others, and as the document progresses the research becomes more focused, ending in the analysis of various works which took on a similar problem to the one that's presented here.

2.1 Privacy

In 2016, the number of Internet users has reached an all-time record of 47% of the world population is using the Internet. When considering only the developed countries, this percentage rises to 81% [7].

With the number of people online increasing, so does the number of dangers to their privacy. In fact, 50% of Internet users cite concern with the amount of information that is available about them online, and 55% have taken steps to escape observation from external entities [6]. Protecting their privacy when using the Internet is, in fact, a matter of paramount importance for many people.

One of the most common measures to ensure confidentiality is the encryption of communications, which is widespread nowadays. Many websites offer HTTPS, which is a protocol for secure data transmission in computer networks. At the same time, data sent wirelessly between a device and an access point is encrypted, making it difficult for an attacker to obtain information. However, there is one type of attack that can circumvent all of these security measures and still obtain information: side-channel attacks. This type of attack, which is not well known to the general public, poses a serious threat to the privacy of many people. In the "Side-channel attacks" section, which can be found some pages ahead, the reason why they are so effective is explained.

2.2 WiFi

WiFi is a wireless networking technology based on the standards described in IEEE 802.11 [8]. Common WiFi networks provide connection to the Internet via an access point, to which a user

can connect to using a range of devices, namely smartphones and laptops.

Since its inception in 1997, the 802.11 standard has gone through several improvements. For instance, one of the most recent revisions, known as 802.11, allows for greater data rates as well as use of multiple antennas.

Today, one can find a WiFi network nearly everywhere in big cities, be it in households, places such as universities and libraries, or even public transportation. They provide fast, reliable access to the Internet and are used daily by millions of people for many purposes. But for for all their strengths, they also have weaknesses, which will be addressed in this section. In addition, we will delve into some of the technical details of WiFi technology that make side-channel attacks possible.

2.2.1 Technical details

802.11 networks are different from their older, fully wired counterpart: Ethernet. The major difference and improvement is the fact that they are wireless, making them a type of WLAN (Wireless Local Area Network). One of the components that makes wireless connection possible is called "access point", or AP, and it is present in most WiFi networks. The AP is responsible for converting the frames used in wireless transmissions into a form that is compatible with the wired part of the network.

RF transmissions generally occur in the 2.4 GHz frequency band [9], but other bands such as 3.6, 5 or 60 GHz are also supported by various revisions of the standard.

WiFi operates in the PHY and MAC layers of the OSI model (figure 2.1). The first handles the physical component, needed for data transmission and reception, while the latter defines they way to access the medium and transfer data.

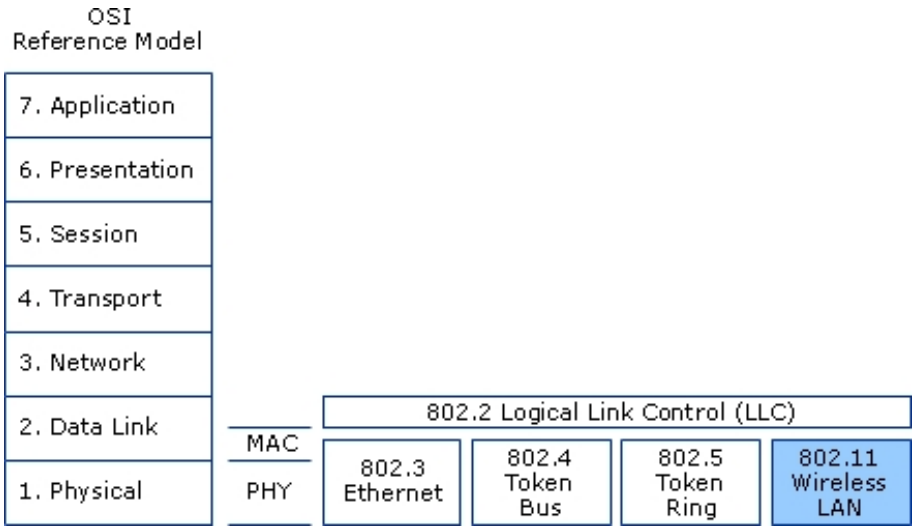


Figure 2.1: How 802.11 fits in the OSI model. (source: [1])

The frames transmitted in the MAC layer transport follow the philosophy of encapsulation present in the OSI model, by adding information relevant to its layer (in broader terms, the link

layer) on top of the data it intends to transport. This additional information is located in the header, which occupies 30 bytes and is divided into several parts.

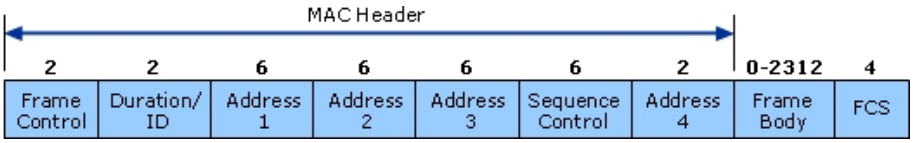


Figure 2.2: The structure of an 802.11 MAC frame. (source: [1])

The structure of these frames includes the following components in the header section:

- Frame Control: specifies the version of the protocol being used. Also includes other information, such as the type and function of the frame.
- Duration/ID: can serve many functions. For example, it can contain the number of a milliseconds the system expects to take to transmit the current frame.
- Addresses: contain the addresses of the sender, the receiver, the source and the transmitter.
- Sequence Control: includes the number attributed to the frame.

After the header comes the frame body, which contains the bulk of the data. The length is variable, since it allows for any amount of data to be transmitted so long as it doesn't exceed the maximum size of the frame. This data is almost invariably encrypted, so as to protect the privacy of the users of the network.

Encryption in WiFi network used to be performed using the Wired Equivalent Privacy algorithm, or WEP, which was the standard until 2004. It used the RC4 stream cypher to provide confidentiality [9]. Its many design flaws made it insecure, and in fact it is possible to crack WEP-encrypted communications with relative ease. The need for a stronger security mechanism facilitated the development of WPA and WPA2, which do not have as many flaws as their predecessor. WPA includes improvements such as message integrity checks and larger security keys [10], while WPA2 - the superior, more recent version - offers further improvements. The most notable ones are the use of AES algorithms, as well as the introduction of CCMP (Counter Cipher Mode with Block Chaining Message Authentication Code Protocol).

2.3 Machine learning

Machine learning (ML) algorithms allow the computer that is running them to perform tasks without being clearly programmed to do so [11]. They are becoming more and more popular, due to their ability to process great amounts of data in order to reach decisions. These decisions are not completely deterministic, as with traditional computer programs. Instead, they use an heuristic approach, which does not yield a completely accurate decision. However, the trade-off is worth it since they allow much more flexibility than traditional programs, while being easier to configure and develop.

An important feature of ML algorithms is that they can learn from data they are "fed". The more data they have, the more accurately they can make their predictions. This is why they are important to this thesis, as we will see ahead.

2.4 Side-channel attacks

As far as encryption goes, side-channel attacks are somewhat different from most commonly known attack. They are not based on flaws in the encryption algorithm or its implementation, and they don't use brute force. In fact, decryption is not an objective of a side-channel attack. Instead, the general approach when performing this type of attack is to obtain side-channel information, which can be defined as the data one can obtain by observing the physical processes in the cryptographic system.

Side-channel attacks are often "more effective than the conventional mathematical analysis based attacks and are much more practical to mount"[12], making them a powerful tool for modern attackers. They can be classified in a variety of ways. For instance, there can be *passive* and *active* attacks, with the former having the attacker in an observing, non-interfering role and the latter implying some sort of forced physical interaction by the attacker. *Passive* attacks are normally *non-invasive*, which means that they exploit that which can be obtained without tampering or sabotaging the physical system. A very advantageous characteristic of this type of attack is that it is virtually undetectable. The attacks which are more relevant to the thesis are, in fact, *non-invasive*. Further ahead we will see some examples.

There are other ways of categorizing side-channel attacks, depending on the exploited characteristics of the target system. Some examples:

- Timing attacks. These attacks rely on the time interval between certain actions in the system. A general activity a user may be performing normally entails a certain sequence of actions which will presumably have a similar timing between them. This allows the attacker to profile the actions and identify them.
- Power Analysis attacks. The computation associated with a process will have potentially similar power consumption profiles every time, providing for a way to identify actions.
- Electromagnetic attacks. EM emissions caused by a given process may be a way to identify them. This type of attack can be very effective and is often used in military contexts. [12]

Having analysed the various characteristics of side-channel attacks, the logical conclusion is that no matter how strong or well-designed an algorithm is, if the system that it will be implemented on is not secure, then it will be vulnerable. WiFi networks are an example of such systems, because a passive observer is able to capture the information transmitted between a device and an access point. The kind of attacks which will be studied for this thesis primarily fall under the "Timing attack" classification, though many of their elements and criteria are not typically considered in the general definition of a side-channel attack.

2.5 Implementations of side-channel attacks

In this section a number of different approaches to side-channel attacks will be presented. Most of them are closely related to the problem at hand (WiFi networks), while some are broader in scope. All of them present considerable insight into what an efficient, appropriate solution might be.

2.5.1 In WiFi networks

Before delving into the many articles written regarding this are, it would be useful to understand the general situation in which side-channel attacks in WiFi networks operate.

Generally speaking, there is a person using a device such as a laptop or smartphone, which is connected to the Internet via an access point. The connection is wireless and encrypted. Nearby, an attacker passively "observes" the traffic between the device and the access point, and then uses the gathered data for a number of purposes, which could include blackmailing or *phishing*. The situation is best understood by analysing figure 2.3 .

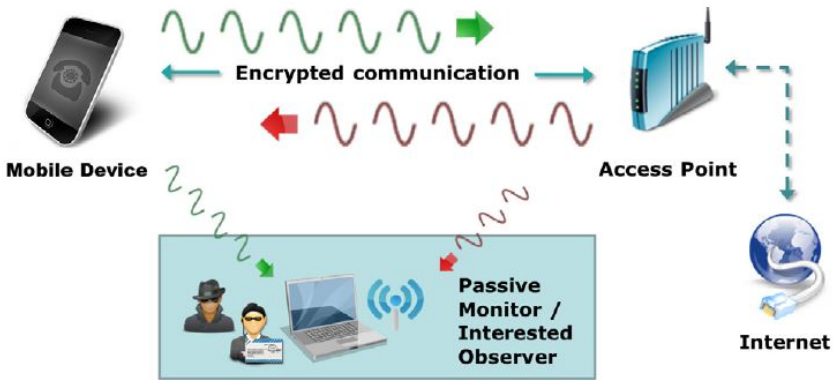


Figure 2.3: A typical side-channel attack in a WiFi network. (source: [2])

As we can see, the target of the attack has no way of knowing they are being attacked, and will probably think they are safe given the encrypted nature of the traffic. However, an attacker can gather a lot of information from this position.

There are some very powerful packet "sniffing" tools, such as Wireshark, that gather information such as the timing, length or frequency of the packets. From a capture file there can be many conclusions, mainly in terms of statistical treatment of the data. The parameters considered in the statistics are called *features*. Which ones the attacker chooses to perform the attack depend on the final objective.

A good example of a side-channel attack in WiFi networks is [3]. The authors developed an algorithm capable of determing which app a given person was using on their smartphone, with only of a few seconds of capture time. They achieved this by using a Machine Learning algorithm, which was fed with data they captured from different mobile applications.

The packet captures were initially done using *Wireshark* in a controlled environment. This is needed in order to have reference points, by being able to associate certain patterns to the mobile

applications. These patterns were then studied, and some features were selected based on how different they were between different apps. The most important features are the frame length and their inter-arrival time.

These captures were then used to develop a Machine Learning algorithm that is able to classify traffic, that is, to determine to which app it belongs to, based on the data it already has. The algorithm is based on random forests (RF), which form decision trees in order to reach a decision. RF are considered to be very efficient, since there are multiple decision trees involved in a decision. This factor decreases the bias and the errors caused by small deviations.

By analysing the dataset they created, it was possible to conclude that applications generate distinguishable traffic patterns, especially when they belong to different categories (for instance, gaming vs. instant messages). This is the essential factor that allows this attack to be effective.

The approach they took for the attack is similar to the one previously described, with a few differences. For instance, they focused only on smartphones, and considered the need to identify a particular smartphone when there are many of them connected to the same access point. The method they used is called MAC address filtering, which takes advantage of certain patterns in the MAC address of the devices. For example, iPhones have similar address structure between themselves. The lists that contain this data are called OUI (organizationally unique identifier) lists.

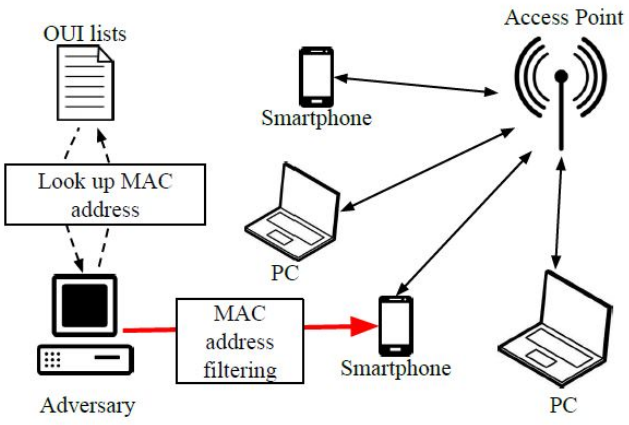


Figure 2.4: Setup used in [3].

After picking a target, the attacker then collects traffic from that user for 300 seconds. This is enough time to determine which application us being used, with accuracy levels no lower than 87% when contemplating the complete features. As we can see, not much time of "attacking" is needed to obtain very good results.

Like the example previously explained, there are other approaches very much worth considering. Another interesting article is [13]. The approach the authors took is quite detailed and varied, allowing for a better understanding of the several ways of approaching the problem at hand. They

used a combination of features to classify the traffic while considering the possibility of concurrent activity between different types of application. The packet capture only has to run for a few seconds for the algorithm to work properly. Unlike [3], the devices used on the testing are laptops.

Among the other articles found regarding side-channel attacks in WiFi networks, [14] and [15] stand out, by covering side-channel attacks in other contexts. In [14] the approach is slightly different since the study is made on 3G networks instead of Wi-Fi. That said, the methodology and approach are quite relevant to this thesis. On the other hand, [15] does use Wi-Fi networks and is relevant for other reasons. While the aforementioned articles have the objective of identifying either the application or the website a person is using, this one aims to classify specific user actions inside some select websites. The information they extract is from websites that handle sensitive information, related to areas such as health or personal finance. This may prove interesting in the perspective of an attacker, seeing that this kind of detailed information may make extortions or phishing scams easier. Similarly, [16] have implemented an algorithm that can detect what a user is searching, based on information leaked by suggest boxes that are common in many websites.

Side-channel attacks can also have a more focused objective, which can make them more accurate. That is the case with [17], in which an algorithm is developed with the ability to detect Skype traffic with very high accuracy.

One of the most potentially damaging models for a side-channel attack was devised in [2]. Their algorithm is capable of extracting very personal information, such as age, gender, religion and sexual orientation. This information is leaked by mobile applications, and can be obtained in real time. The drawback is that the false positive rate is relatively high: 38%. That said, it is still a highly dangerous attack, that if improved represents a very serious risk to privacy in WiFi networks.

2.5.2 Others

Besides the types of side-channel discussed in the previous section, there are others worth mentioning, even if they don't have as much of a direct relation to the approach described in this report.

Side-channel attacks on smartphones are not limited to WiFi networks. In fact, it is possible to capture data transmitted through 3G/UMTS - and in [14], that is the approach they took. The final objective of their investigation was to determine if it was possible to fingerprint a specific device based on the background traffic generated by the apps it is running, and indeed it was. To that end, captures are made to obtain the relevant side-channel information, which is then classified according to features such as timing or bytes transmitted. Based on this, the algorithm is capable of associating a phone with a specific UMTS cell, which represents a privacy breach. The main concern about this approach is that the capture time needs to be of about 15 minutes in order to obtain optimal accuracy (more than 90

Another side-channel attack which can represent a very serious invasion of privacy is [18]. Their algorithm uses leaked electromagnetic information to deduce a mobile phone's password

with a high accuracy rate. When holding a phone, the way the hand is holding it and the finger motions used to input the password lead to an interference to multi-path signals. These are reflected by the CSI, or channel state information. The algorithm, when in possession of this information, can "exploit the strong correlation between the CSI fluctuation and the keystrokes to infer the user's number input". After obtaining the password, the attacker may be able to compromise the phone and the accounts associated to its apps.

2.5.3 Counter-measures

Defending against a side-channel attack can be a challenging task. As we saw, these attacks bypass any attempts at encrypting the communications between the user and the access point. To make things worse, normally speaking a website or mobile application does not take any precautionary measures to defend against side-channel attacks.

The communication between a device and the AP is done when necessary and as soon as possible, which allows for *timing* attacks. On the other hand, each frame sent or received contains only the data it needs to transmit, which permits attacks based on the size of the packets. With this in mind, an effective counter-measure would be to time the frame transmission in regular intervals regardless of the content, as well as distribute the data throughout several frames. This solution entails some problems however, mainly in the aspect of user experience. Instead of the quasi-instantaneous communication a modern Internet user has come to expect, because of these limitations many delays are created, which severely compromises the responsiveness of the app or website.

An interesting solution, described in [19], involves traffic demultiplexing in the MAC layer. Unlike the solutions previously described, it does not cause any significant drop in performance, and the overhead is small. Using packet scheduling and MAC layer virtualization, it is possible to mask the characteristics of the traffic, thus making side-channel attacks very difficult.

Another solution is using a covert channel to transmit data. In [20], the authors developed a protocol that uses rate switching to hide data within an existing communication channel. However, this solution is dependant on the exploited system being non-deterministic or noisy "since deterministic protocols are certifiably free of covert channels".

2.6 Analysis

Most publications about side-channel attacks do not consider the influence of network-level noise on the accuracy of the classification algorithm. Instead, they test the actions - which can be accessing a website or using a program - on a vacuum of sorts, in which it is assumed that no other traffic beside the one that is the object of study is being generated, though some studies do consider the possibility of noise coming from concurrent applications or programs running simultaneously, like for example [3]. Analyzing the effect different traffic volumes and types of network-level noise have on the classification is something that has not been studied in depth yet.

The analysis of the differences between static and dynamic websites as far as the accuracy of traffic classification goes is another under-explored theme. Detection accuracy may be linked to how many dynamic, or interactive elements websites have. These elements can include videos, banners with live information and advertising. There is no clear data indicating that one type of website is easier to detect that the other, or if a classification algorithm can differentiate between an dynamic or a static website.

After processing traffic captures, the normal procedure is to calculate a set of features to help distinguish them. These features are used by the detection algorithm, which often makes use of machine learning, to predict and classify new captures. This means that they are the basis for most, if not all, side-channel attacks on 802.11. However, these features normally do not characterize a traffic capture according to the shape of its traffic. The traffic can have unique traits that makes it easier to be identified, as long as a feature which represents those characteristics is used.

These traffic shapes can be represented through graphs that plot the volume of data in bytes transferred over time. Algorithms such as Dynamic Time Warping [21] could be applied so as to determine mathematically how similar any pair of them are.

Chapter 3

Side-Channel Attack

The performing of a side-channel attack by an attacker requires preparation, including a theoretical approach appropriate to the attack scenario. In this chapter, both aspects are studied.

3.1 Problem description

In this section the main problem, which this project aims to solve, will be explained.

3.1.1 Introduction

For the purpose of this dissertation we will assume the role of a potential attacker. In order to understand both his objectives and his potential limitations, it is useful to describe the scenario in which the attack would unfold.

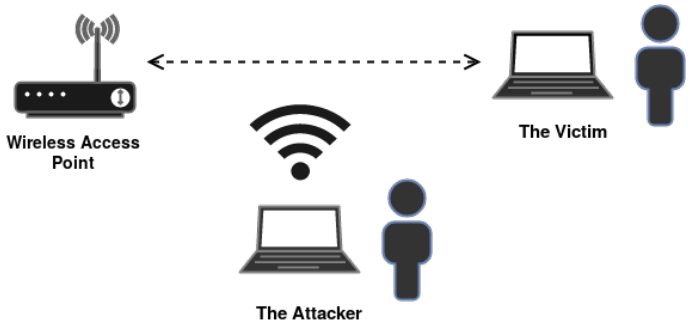


Figure 3.1: Scenario of a potential attack

First and foremost, we have the victim. In this case, the victim could be any unsuspecting person using their laptop to browse the Web, as long as it's connected to a WiFi network. Nowadays most of these networks offer WPA encryption, so we can assume that the traffic to and from the device would be encrypted and theoretically indecipherable to others.

The attacker would be positioned within a few meters. Using his laptop, he would attempt to capture the traffic flowing between the victim's own laptop and the access point. To do so the attacker does not need to be connected to the same network as the victim - in fact, he probably

wouldn't. So as to perform the captures the laptop's wireless network adapter would have to be configured in monitor mode, which in most models means that it can only monitor traffic, thus disabling normal functioning and rendering the attacker unable to do some internet browsing of his own. This limitation can be circumvented if the laptop has more than one WiFi interface - one could be set to capture traffic, while the other is reserved for other activities.

This attack could be performed without alerting the victim, who would have no way to realize that it's underway. In fact, the attacker assumes a primarily passive role. No direct privacy breaches are attempted, such as gaining access to the victim's device or decrypting the traffic.

3.1.2 Attack model

After having captured some traffic, the attacker would then try to extract information from it. For the sake of this project, we assume that they would seek to identify which website the victim had connected to. All the data would be encrypted, so the attacker would have to find a way to perform website identification without decrypting anything. As we have seen before, this would be a type of side-channel attack.

It is possible to process the data contained in the capture files so as to obtain "features", which could be used to identify the websites. In order to do that the attacker would need to have a database of features for the websites he wishes to be able to detect, which represents a limitation. If the victim accesses a website for which the attacker didn't previously account for, the identification would either be impossible or result in a false positive.

3.1.3 Influence of network-level noise

The scenario described thus far represents a quasi-perfect situation, in which there are no external factors conditioning the effectiveness of the attack. It assumes that the victim would access a single website, without any other activity being performed in parallel. This does not represent an entirely realistic scenario, as more often that not when someone is browsing the internet there is some other program transferring data. For instance, music streaming services and *torrenting* can be running in the background. Even if the victim isn't running any application that generates this kind of traffic there are still some types of network traffic that normally occur like DHCP, which is related to the management and attribution of IP addresses, interfaces and other network-related configurations, or ARQ, a protocol that handles errors in data transmission. We can classify this type of data transfer as network-level noise.

Network-level noise represents a setback for the attacker. All communications are wrapped under 802.11 encryption protocols, which means that website detection could be hampered or indeed nullified if the noise is sufficiently disruptive. Therefore, it is important for a potential attacker to take these factors into account, and make sure their detection method is as effective as possible even when faced with situations in which there is substantial noise. For that purpose, the attacker must extend the database to include features of captures with multiple combinations of websites and network-level noise.

3.2 Solution

A possible solution to the problem presented in the previous section will now be explained. This section focuses on the theoretical elements of the solution, while the details of the implementation are presented in the next chapter.

Figure 3.2 shows the steps the attacker takes in order to successfully perform a side-channel attack.

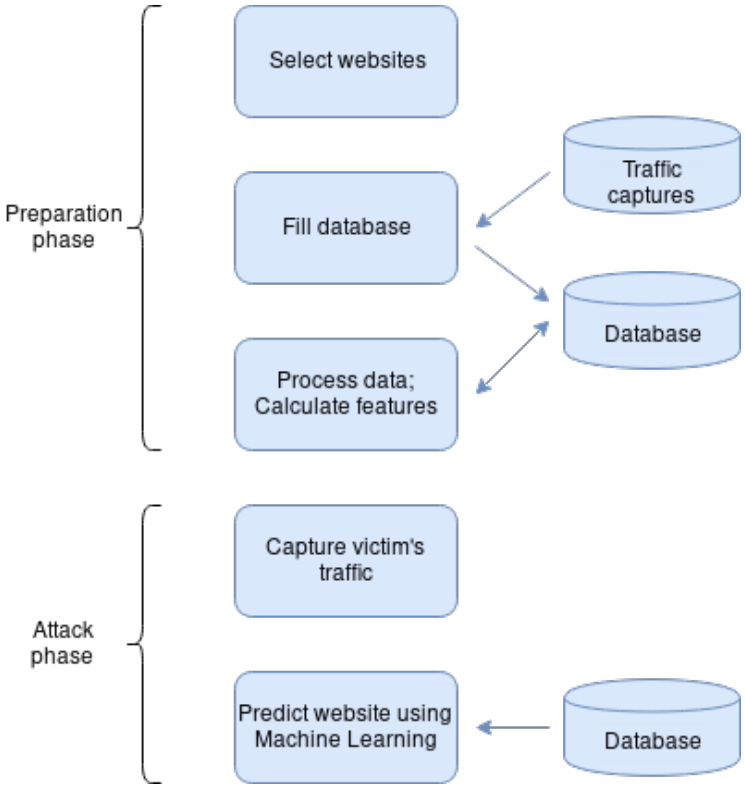


Figure 3.2: Flow of an attack, from the preparation to the execution

3.2.1 Website selection

As it was said in the previous section, the attacker can only detect the websites whose features are stored in their database. This means that an important first step is to decide on a list of websites, based on the objectives of the attack.

For the purpose of this dissertation, several websites where considered. Initially the focus was on social media websites such as Facebook or Twitter, as they are some of the most widely used. However, the look and content of these websites is often determined by the personal preferences and the activity of the user, meaning that a meaningful dataset would be harder to obtain. The experimental setup, which is explained in the next chapter, captures website information in their default status, thus bypassing the influence of user's customization. Capturing website data on social media websites would require the automated capture system to perform a *login* every time,

hence requiring a number of "test accounts". Setting them up would not be very practical, so in the end it was decided that other websites would be used.

Among the alternatives considered, news websites seemed like the best candidates. They are fairly popular, making them a viable target for an attack. Furthermore, their content is generally not dependent on whether the user is logged in or not.

The decision on which websites to use was based on a few factors. First of all, there was the objective of guaranteeing a certain diversity between them, both in terms of content and design. There was also the preference for websites supporting HTTPS, so as to allow the traffic captures to be integrated in another project which involves TLS-based fingerprinting. Finally, their popularity was also taken into consideration, as a potential attack would have a greater chance of success with a more widely known website.

In the end, five websites were chosen:

- Wall Street Journal (wsj.com)
- New York Times (nytimes.com)
- Bloomberg (bloomberg.com)
- The Guardian (theguardian.com)
- Financial Times (ft.com)

After further consideration, a decision to include five more websites was made. The reasoning is that though the previously chosen websites were indeed diverse among themselves, they were all dynamic in the sense that their content could vary greatly between captures. For instance, the advertisement panels change frequently, as well as the front page by consequence of the different news being highlighted. Banners with live updates are also variable according to the time the website is accessed.

The addition of mostly static websites to the final list of websites for testing would help to address the issues raised in the previous paragraphs, as well as verify the level of their influence on the detection algorithm. Thus it was decided that five new websites would be added. They were chosen among the University of Porto's Faculties', with all of them belonging to the *Sigarra* "family". These websites are:

- Faculty of Science (sigarra.up.pt/fcup)
- Faculty of Engineering (sigarra.up.pt/feup)
- Faculty of Law (sigarra.up.pt/fdup)
- Faculty of Arts (sigarra.up.pt/flup)
- Faculty of Medicine (sigarra.up.pt/fmup)

The final list includes ten websites, of which half can be classified as static and the other half as dynamic. This characterization is not completely accurate in the sense that all websites include both static and dynamic elements. However, there is a clear difference in the two groups.

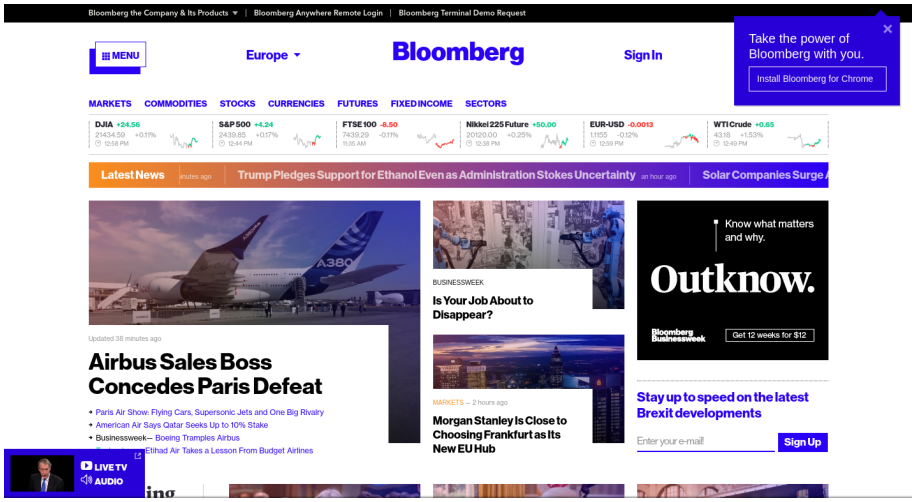


Figure 3.3: Screenshot of *bloomberg.com*

Figure 3.4: Screenshot of *sigarra.up.pt/feup*

Take the website *bloomberg.com* for example. As we can see on figure 3.3, there are some elements that can be considered dynamic. There is a video playing on the lower left-hand corner, a banner with graphs showing the evolution of stocks, and an advertisement on the right-hand side. And though it cannot be seen on the screenshot, some of the images on the background of the news' titles are animated.

On the other hand, a more static website such as FEUP's, which can be seen on figure 3.4, does not have any of these elements. The banners are static, and there are no animated advertisements or videos playing. Therefore, it was decided that the manner in which these differences may result in different detection accuracy rates would be studied.

3.2.2 Features

Having selected the websites, the next step is finding the best way to characterize them. The method to do so involves the calculation of features, which are derived from the processing of traffic capture logs.

3.2.2.1 Raw data

The traffic capture logs contain information regarding to each packet, such as the MAC address of both the origin and the destination, the *timestamp* and the size in bytes. Some of these features can be seen in 3.5, which is a screenshot of a part of a traffic log visualized on the Wireshark software. For instance, from left to right one can see the frame number, the relative *timestamp*, the source and the destination. Other information that is not relevant is also present, though it can be filtered on the data processing phase. That is explained on the next chapter.

```
289 7.748774  Tp-LinkT_0f:27:e0        Tp-LinkT_0e:62:f1        802.11   1566 Data, SN=3452, FN=0, Flags=.p....F.
290 7.749091  Tp-LinkT_0f:27:e0        Tp-LinkT_0e:62:f1        802.11   1566 Data, SN=3453, FN=0, Flags=.p....F.
291 7.749726  Tp-LinkT_0f:27:e0        Tp-LinkT_0e:62:f1        802.11   1566 Data, SN=3455, FN=0, Flags=.p....F.
292 7.750373  Tp-LinkT_0f:27:e0        Tp-LinkT_0e:62:f1        802.11   1566 Data, SN=3457, FN=0, Flags=.p....F.
293 7.750517  Tp-LinkT_0e:62:f1        Tp-LinkT_0f:27:e0        802.11    118 Data, SN=182, FN=0, Flags=.p.....T
294 7.750613  Tp-LinkT_0e:62:f1        Tp-LinkT_0f:27:e0        802.11    118 Data, SN=183, FN=0, Flags=.p.....T
295 7.750721  Tp-LinkT_0e:62:f1        Tp-LinkT_0f:27:e0        802.11    118 Data, SN=184, FN=0, Flags=.p.....T
296 7.770671  Tp-LinkT_0e:62:f1        Tp-LinkT_0f:27:e0        802.11    126 Data, SN=185, FN=0, Flags=.p.....T
297 7.773164  Tp-LinkT_0e:62:f1        Tp-LinkT_0f:27:e0        802.11    129 Data, SN=187, FN=0, Flags=.p.....T
298 7.774245  Tp-LinkT_0e:62:f1        Tp-LinkT_0f:27:e0        802.11    130 Data, SN=188, FN=0, Flags=.p.....T
299 7.778639  Tp-LinkT_0e:62:f1        Tp-LinkT_0f:27:e0        802.11    118 Data, SN=189, FN=0, Flags=.p.....T
300 7.786735  Tp-LinkT_0e:62:f1        Tp-LinkT_0f:27:e0        802.11    526 Data, SN=190, FN=0, Flags=.p.....T
```

Figure 3.5: Excerpt of a traffic capture log

Each packet also includes a set of flags. From those flags it is possible to infer whether a packet is a *Retry* or not. A *Retry* packet is a copy of another packet that did not reach its destination. The 802.11 protocol supports the *Retry* feature in order to ensure that all packets are properly sent and received.

These packets are not relevant to the attacker, since they are generated as a result of circumstances outside of the victim's direct actions. For instance, the greater the distance of the victim's laptop to the access points the more of these packets will be present, due to the poorer connection. Since *Retry* frames are easily identifiable through a flag, the attacker can filter them and therefore not consider them when processing the captures.

All packets are encrypted by the WPA protocol, present in most WiFi networks. Therefore, the amount of relevant information one can extract from each individual packet is low. However, it is possible to analyze the entire log and extract features which might make distinguishing the websites possible. A total of seven features were chosen, based on calculations with the information that can be extracted from an encrypted capture.

3.2.2.2 Byte, packet and bitrate-based features

The five features presented in this section are based on the number of packets sent, as well as their size in bytes. These are features are:

- Byte count: The accumulated size of the packets, in bytes.
- Packet count: The total number of packets.
- Average packet size: The average size of a packet, in bytes.
- Average Mbytes/s: The average speed of data transfer.
- Average Packets/s: Average number of packets sent each second.

3.2.2.3 Burst rate

The features presented on the previous section do not take into account variations in the *bitrate* and *packet per second* rate. These can be important in defining a website, since each one generates a different traffic pattern. For example, in 3.6 one can see an excerpt of a traffic capture, represented through the packets sent every ten milliseconds. The shape of the pattern is not reflected in any way in any of the previous features, since they just use the total values for the calculations.

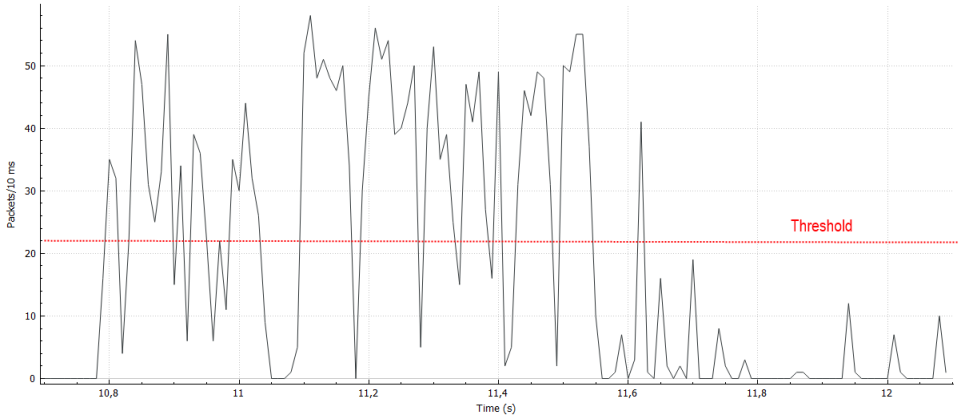


Figure 3.6: Portion of a traffic capture graph

On that same figure one can also see that the traffic has periods in which it peaks, forming a peak or a *burst*. These bursts can also be a way of characterizing a website, and so it became relevant to design features that would take the *burstiness* of the capture into consideration.

These new features are:

- Burst rate: Corresponds to the percentage of the capture that is considered part of a burst.
- Average of the top 3 *packets sent per window*: The average of the three highest number of packets sent on a 10 millisecond window.

In order to calculate the burst rate, the capture is first divided into intervals of ten milliseconds. The value of ten milliseconds for the window provides sufficient resolution while not being too taxing on the processor. On figure 3.6 one can visualize that this window is enough to see bursts. For each interval, the number of non-retry packets are counted and then stored onto an array. The

next step is to calculate the average of the three highest values. This average is one of the features, and it permits the distinction between websites when the average "height" of their three tallest bursts is different. It assumes that the same website will, for the most part, generate bursts of similar height across captures.

A threshold is defined - in this case 40% of the aforementioned average - and when one of the intervals has a value that exceeds it then it can be considered as being part of a *burst*. On figure 3.6, all intervals in which the line is above the red threshold are bursts.

The burst rate, which is the final feature, represents the level of *burstiness* of a capture. It can be calculated as the percentage of intervals present in the capture that can be considered *bursts*, as defined in the previous paragraph. In the case of two websites with a similar byte and packet count, this feature can distinguish them, because the traffic shape can potentially be very different.

3.2.3 Dataset structure

The database is an important element in the structure of the attack. It is there that all the relevant information regarding each captures is stored, to be later used as tool for identification. Therefore, it is important that the database is structured in a functional way. For this proposed solution, the database would be divided in three different tables: one for the raw, relevant information for each packet of each capture; one for the features of each capture; finally, one for statistics regarding each website.

The statistics would not be used for the attack. Instead, they would provide a visualization of the average value of each feature, as well as its standard deviation, divided by website. Ideally, the average values would be distinct for each one. A low deviation is also desirable, because the more unpredictable a website it the more difficult it becomes to identify it.

3.2.4 Machine learning algorithm

Having a dataset is not enough to be able to identify websites. In fact, that detection would be performed through a machine learning algorithm, which would be able to predict the identity of a new capture based on the previously calculated features.

As a first step, an attacker would need to fill the database with information corresponding to the websites that they would want to be able to identify. This includes, most importantly, the features for each one. A high number of captures for every website would help ensure greater statistical strength by representing possible variations within the same websites, making for a greater chance of success when using the algorithm.

There are many kinds of ML algorithms, which means that finding out which of them work best with the features available is important. For that purpose some testing is required, as accuracy can vary greatly according to which algorithm is chosen.

These are the algorithms and methods that were tested, using the *sklearn* Python package:

- Logistic Regression
- Linear Discriminant Analysis
- K-Neighbors Classifier
- Decision Tree Classifier
- Gaussian Naive Bayes
- Support Vector Machine

From these algorithms, two had a higher projected accuracy when compared to the others: Decision Tree Classifier and Linear Discrimination Analysis. Tests show that the value of the accuracy was above 60%, while other algorithms performed below 50%. . Further testing showed that the Decision Tree Classifier yields slightly better accuracy, and was thus chosen for the next phases of the testing.

Having completed the aforementioned preparations, the potential attacker might use the values stored in the database to *train* the algorithm. When "fed" with the features of a new, unidentified capture, the algorithm would perform a prediction.

Chapter 4

Methodology and Approach

This chapter describes the process of taking the concepts and ideas described in the theoretical solution and applying them into an experimental setup and procedure.

4.1 Experimental setup

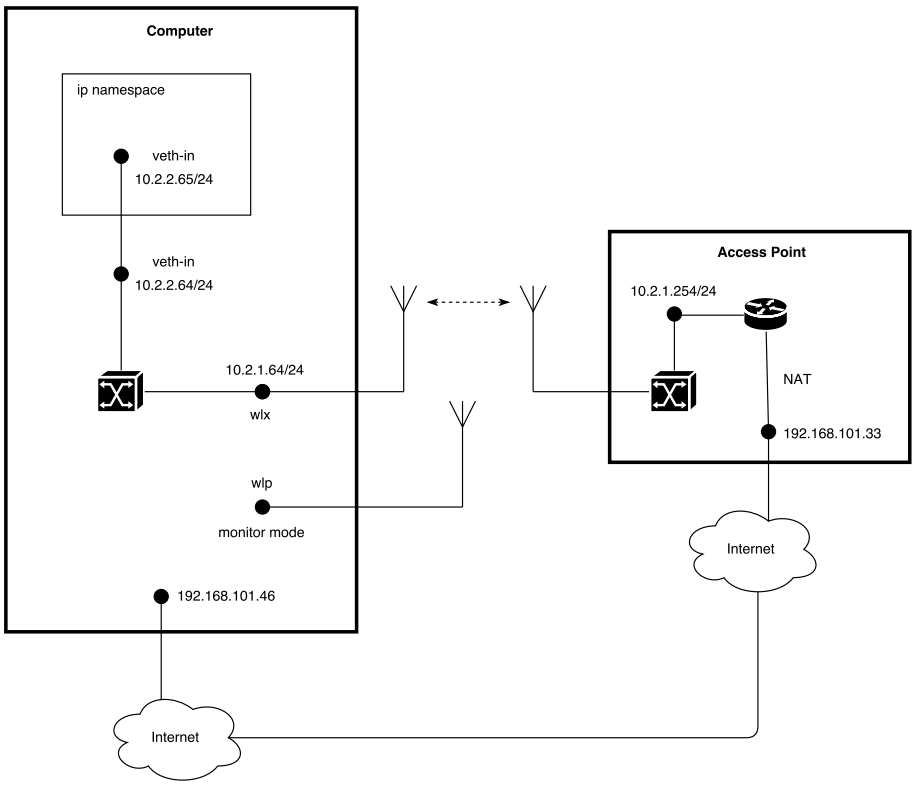


Figure 4.1: Diagram of the various elements of the experimental setup

There are two main elements in the experimental setup: a laptop computer with a WiFi-capable network card, and an access point connected to the internet. The connection between these two is made through the WiFi network generated by the AP.

The access point, which can be seen on the right-hand size of figure 4.1, is a router connected to FEUP's network which also provides access to the Internet. The reasoning behind the inclusion of a personalized access point instead of using one of the existing ones is to ensure full control of the network, therefore avoiding the interference of outside factors. The router can handle two simultaneous connections coming form the same computer, as we will see further ahead. Additionally, outside connections coming from other computers are allowed, so as to permit the remote operation of the laptop computer through an SSH connection. These connections are handled through a cable connection, unlike the rest of the experiment.

The laptop possesses a network card, but for the purpose of the experiments an extra wireless-capable network card is connected to the computer. The first card is used for connecting to the Internet, while the second one is set to *monitor mode*, which allows it to simulate the perspective of an attacker and obtain WPA-encrypted captures similar to what would be obtained in a real-world setting. This way, the need for an extra computer to perform experiments is eliminated, as a single laptop handles both the automated website access and the capture of its traffic.

The *wlx* interface that we can see on figure 4.1 handles the connection to the AP and, therefore, the Internet. The traffic generated by accessing the website flows through there. The *wlp* interface, on the other hand, is set to monitor mode and doesn't have an assigned IP address. The aforementioned traffic is captured by this interface, using *tcpdump*, and saved onto a file.

However, there is a drawback to this setup, since the computer being connected to the Internet via cable represents a problem. In normal conditions, applications and programs such as browsers would, by default, use the cable connection instead of the wireless one, meaning that capturing traffic becomes impossible. For the purpose of avoiding this situation, a solution was found: creating a network *namespace*, inside which the browser and other programs related to traffic capture would be launched. Executing these applications inside the *namespace* means that they are "unaware" of all interfaces except the wireless one which they are supposed to use. This interface connects to the Internet through the router, bypassing the cable connection and therefore allowing the interface in monitor mode to capture the traffic needed for the experiments. In parallel, remote control of the laptop is done through the cable connection, thus avoiding interference with the experiments.

The experimental procedure, which works differently when noise is supported, will be described in the next sections. The same hardware is always used - however, generating and accounting for noise requires further software.

4.2 Capturing traffic

Capturing traffic involves the use of several *Python* scripts, supported by the previously described hardware as well as a number of programs and drivers.

4.2.1 Without network-level noise

Before performing a capture without noise, it is first necessary to ensure that all interfaces are set correctly, as well as ensuring connectivity. After the initial check-up, the following sequence ensues:

1. Using *tcpdump*, a program that allows capturing traffic, a new capture is initiated through the interface that is set in monitor mode. This is necessarily the first step, so as to ensure that everything is properly captured in the log file. The command used to start the capture includes a filter that limits the packets it saves to the ones associated with the laptop's MAC address. This is followed by a one second interval, so as to allow the program to properly start.

2. The wireless connection to the access point is reset, by means of two *nmcli* commands. The reason behind this step is that in order to decrypt the capture file, thus removing the 802.11-layer encryption, it is necessary to capture the 4-way *handshake* used by the WPA protocol to negotiate a shared encryption key. If one has access to the network's SSID and password, it is possible to fully decrypt the capture file.

3. Since resetting the connection also deletes an important *route* for the namespace, without which it cannot connect to the access point, a command is used to re-add this route.

4. Using Selenium, an instance of the Google Chrome browser is launched from within the *namespace*, so as to force connection through wireless. The SSL key logfile is saved at this point, so as to allow for decryption of the TLS layer if necessary. The browser accesses the *URL* that was indicated, and waits for the page to be fully loaded. Afterwards it shuts itself down, signaling the end of the capture.

5. The capture that was initiated in step one terminates itself, saving all the data gathered thus far on a properly name files, allowing for quick identification of both the experiment and website it belongs to and the number of the repetition. All other programs used for the capture are also terminated.

4.2.2 With network-level noise

When simulating the existence of network-level noise it is necessary the emulate it by generating traffic. In this case, UDP traffic with a constant bitrate was used.

UDP traffic was chosen over TCP as a way to generate noise because of its more suitable characteristics. For instance, UDP is a connectionless protocol and therefore does not need or

use acknowledgment packets, meaning that the packets being generated are sent regardless of any network congestion measures usually implemented in TCP. This independece means that UDP is more reliable in its behaviour, especially since there is no interest, in this case, in assuring that every single packet reaches its destination.

The sequence of events for a capture with network-level noise is quite similar to the already presented one, with the addition of two new steps, to be placed between steps 3 and 4 of the previous list. These take advantage of the features of a program called *iperf*, which allows for the generation of TCP or UDP traffic between two points. One of the two instances of *iperf* needs to be set as server, while the other acts as client which creates and sends the packets to the former. For the experiments,

The new steps are:

1. Create an instance of an *iperf* server on the laptop, which is listening for UDP traffic.

2. Create another instance of *iperf*, this time as a client. This instance must be placed "inside" the namespace, so as to ensure the traffic flows through the same components as the rest of the communications - namely that of the browser. The client starts sending UDP traffic at a rate defined by the user, until it receives the shutdown signal after the capture ends.

Four levels of noise were considered: 10 KB/s, 50 KB/s, 100 KB/s and 200 KB/s. The values were chosen so as to emulate both residual noise and traffic at the level that which accessing a website generates, as well as values in between.

The aforementioned bitrates are far from the saturation point of the wireless interface that was used for the experiment, which has a value of 6,75 MB/s. However, traffic generated at values close to that would "drown out" the website access, since the values that were measured show that a capture with no noise has a rate of around 200 KB/s in average. Therefore, the values that were chose allow for a better characterization of the influence of network-level noise on the accuracy of the classification algorithm.

4.3 Data processing

An important factor of the experiment is the processing done on captures after they are saved. *Python* scripts handle this job. When given a folder containing WPA-encrypted capture files, they extract the relevant information and save it onto a database.

For each file, the procedure is the following:

1. The connection with the database is established. If there is no database yet, it is created.

2. Using *Wireshark*, the network packet dissection of the capture is saved in a PDML (Packet Description Markup Language) file. This a file type that supports XML standards, and provides for a convenient way to access and process the raw data of each capture, divided by packet.

3. For each packet in the PDML file, the relevant information is extracted and saved onto a database file. This includes:
   (a) The frame number, which represents the order in which the packet was captured;
   (b) The relative timestamp, which represents the time passed in seconds since the beggining of the capture;
   (c) The size of the frame in bytes;
   (d) The sequence number, which is a count of the frames performed by the 802.11 protocol;
   (e) The MAC address of the source;
   (f) The MAC address of the destination;
   (g) The flag signaling whether the packet is a retry or not.
4. Optionally, a graph of the number of bytes transferred along the total time interval can be generated.
5. The features for the whole capture are calculated, while ignoring the packets flagged as retries. The ratio of retry packets is calculated, though it is not used as feature. The final values are stored on the database.
6. After calculating and storing the features for all the captures, statistics are calculated for each website and stored in the database as well.

Chapter 5

Results

The first step before obtaining results is filling the database with captures and calculating their features. Afterwards, *Python* scripts are used to perform testing on the classification algorithm, as well as generate graphs, matrices and tables to help visualize the results. The tests are mainly focused on measuring the accuracy of the machine learning prediction.

5.1 Dataset preparation

For each of the ten websites and for each of the five levels of noise (including the case in which there is no noise), 25 captures were made, so as to confer statistical significance to the results. Following the procedures described on the previous chapter, captures were performed in automated fashion over the course of a few hours.

There were a few captures that were not successful, due to crashes in the *selenium* drivers. These captures were filtered out so as to avoid confusion on the detection process, because the features calculated for these captures would be skewed when compared to the average of the successful captures.

Following this, the captures were processed and its features were calculated and stored in the database.

5.2 Preliminary analysis

5.2.1 Plots and graphs

Before testing the prediction algorithm it is possible to investigate the differences in the websites by plotting the graphs corresponding to their captures and analyzing them. The most obvious way of doing that is through empirical observation of the graphs, since the patterns generated by each website have noticeable differences. The graphs generated for initial analysis are an accumulated time series of the bytecount transmitted throughout the capture.

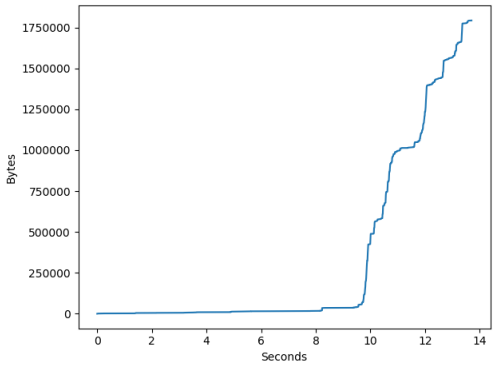


Figure 5.1: Accumulated time series of a *ft.com* capture

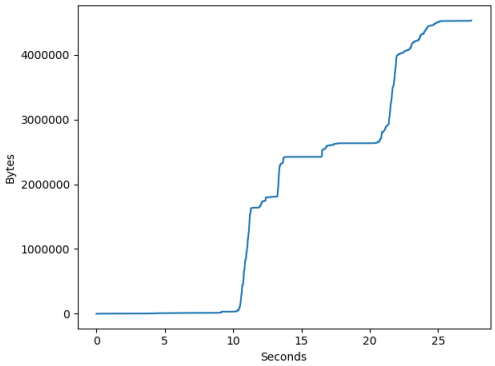
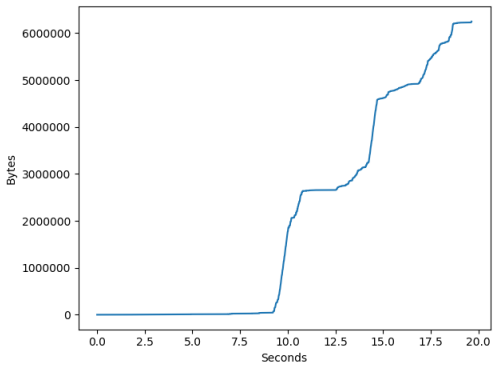


Figure 5.2: Accumulated time series of a *theguardian.com* capture



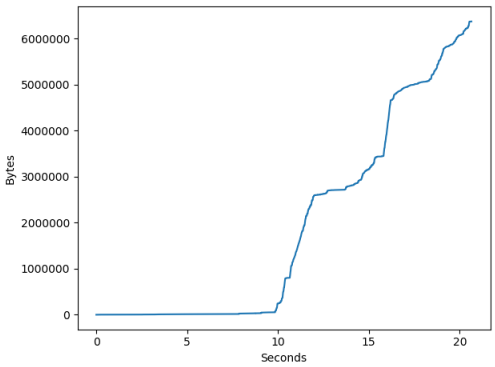
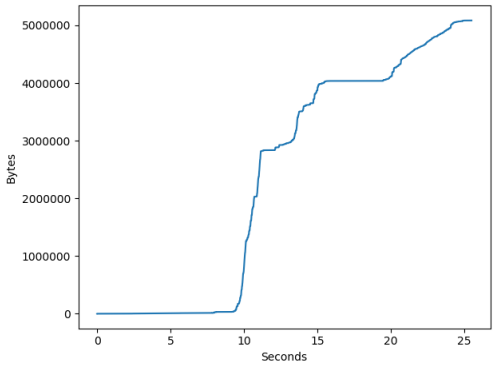


Figure 5.3: Accumulated time series of two *wsj.com* captures



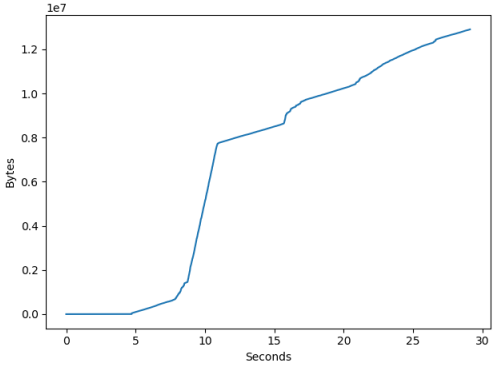


Figure 5.4: Accumulated time series of two *bloomberg.com* captures. The left one has no network-level noise, while the one on the right has a noise level of 200KB/s

Figures 5.1 and 5.2 show the accumulate time series for two websites: Financial Times and The Guardian. The total number of bytes transmitted are displayed on the vertical axis, while the seconds passed are represented the horizontal axis. By observing the two graphs, one can see that the shape of the traffic is noticeably different between the two websites. While accessing *ft.com* results in the bulk of the data being transmitted within a smaller time frame, accessing *theguardian.com* results on a more prolonged and irregular loading of the data. On the latter we can see that there are gaps of around two seconds in which no data is loaded, resulting on a step-like traffic shape.

On figure 5.4, one can verify that the presence of constant network-level noise influences the shape of the traffic pattern. The variations in the byte count become more diluted because of the noise, which in a way smooths the graph. If the noise was great enough in volume, the graphs of all websites would be come all but impossible to distinguish visually.

Besides this empirical analysis, one can use Dynamic Time Warp to mathematically compute a value representing the similarity between the shapes of two graphs. A *Python* library called *fastdtw*, which is based on [21], was used to perform tests. Applying DTW to two time series yields a distance. There are many types of formulas that can calculate that distance. The one used for the tests in this chapter is the Bray-Curtis dissimilarity, given by

$$\frac{\sum |u_i - v_i|}{\sum |u_i + v_i|} \tag{5.1}$$

where u and v are two-dimensional arrays.

For example, the distance between the two captures displayed in figures 5.2 and 5.1 is 1160 (rounded to the units). But if we take two captures of the same website, like the ones in 5.3, the distance is only 3.

If we apply DTW to the two time series of figure 5.4, which belong to the same website with different noise levels, the result is a distance of 2221. This value is higher than the one described in the previous paragraph for two different websites, which might seem unexpected. It seems to indicate that noise has a great impact on the detection of a website. However, the features that are used to classify a website can be independent from this fact.

5.2.2 Statistics of the features

The classification algorithm uses the features of every capture in order to perform a prediction. However, the database also includes a "statistics" table, which contains both the average and the standard deviation of the features for each site. Analyzing this information, which is on table 5.1 allows for a preliminary characterization of the websites, and therefore the identification of differences between them.

Table 5.1: Statistics of the features of the captures present in the dataset (without noise)

Website	Parameter	Byte Count	Packet Count	Packet Size	Burst Rate	Top 3 Values	MB/s	Packets/s
ft	average	1413307	2014	701	2.777	42.231	0.105	149.822
	deviation	243826	349.491	14.491	0.575	3.928	0.017	25.260
wsj	average	6073482	8506	714	5.047	63.983	0.282	393.859
	deviation	461261	734.317	26.532	2.003	13.164	0.065	90.733
feup	average	1428462	2545	560	4.820	57.946	0.134	239.382
	deviation	124116	206.601	8.062	0.630	3.991	0.017	28.444
flup	average	3341890	5844	571	9.878	67.472	0.284	497.338
	deviation	70831	117.179	9.486	0.886	4.929	0.023	38.864
theguardian	average	4540632	6427	706	4.174	54.533	0.187	264.547
	deviation	203522	278.522	20.322	1.283	9.857	0.039	53.509
fmup	average	3041932	5271	576	9.006	65.848	0.260	451.633
	deviation	159733	282.223	6	1.006	4.195	0.027	47.625
fcup	average	3022676	5338	565	9.975	68.069	0.286	505.577
	deviation	172181	311.392	10.246	0.934	4.925	0.027	48.387
nytimes	average	4465572	6344	705	4.066	49.173	0.190	270.362
	deviation	385836	694.356	23.366	0.874	7.811	0.023	33.870
bloomberg	average	5102296	7106	718	3.614	56.333	0.175	244.297
	deviation	249274	468.265	24.637	0.931	8.830	0.031	41.686
fdup	average	3691846	6410	575	10.913	69.507	0.321	558.764
	deviation	99798	131.711	7.681	0.920	5.736	0.025	39.147

Table 5.2: Statistics of the features of the captures for *sigarra.up.pt/fmup* present in the dataset, with all noise levels considered

Noise Level	Parameter	Byte Count	Packet Count	Packet Size	Burst Rate	Top 3 Values	MB/s	Packets/s
No Noise	average	3041932	5271	576	9.006	65.848	0.260	451.633
	deviation	159733	282.223	6.00	1.006	4.195	0.027	47.625
10 KB/s	average	3157842	5400	584	8.193	65.869	0.241	413.823
	deviation	148845	271	6.855	0.782	6.391	0.020	36.965
50 KB/s	average	3536767	5708	619	7.780	65.319	0.249	403.505
	deviation	105844	145	12.767	1.150	4.233	0.034	57.484
100 KB/s	average	3923669	5920	662	8.120	65.287	0.285	431.654
	deviation	92466	80	11.618	0.770	4.355	0.023	36.348
200KB/s	average	4530853	6235	725	8.625	61.746	0.341	470.327
	deviation	263824	246	17.435	0.838	4.939	0.021	31.434

The values on table 5.1 show that some differences are apparent. For instance, the byte count of the dynamic websites is, generally speaking, noticeably higher than that of static websites. The average packet size is also higher in dynamic websites, which means than even on this preliminary analysis one can distinguish between the two categories of websites.

By doing a website-by-website comparison one can also identify case in which there is a group with very similar features. That is the case with *flup* (*sigarra.up.pt/flup*), *fmup* (*sigarra.up.pt/fmup*) and *fcup* (*sigarra.up.pt/fcup*), whose features are, as far as average values go, close to each other. Therefore it is predictable that the classification algorithm will have some difficulties in distinguishing the two websites.

The presence of network-level noise on the captures influences the features of the websites. The case for FMUP's website is presented on table 5.2. Analysis of the table shows that, in general, most features increase in value when the volume of noise is higher. This is especially noticeable in the "Byte Count" and "Packet Count" features - which makes sense since the noise introduces more packets, each adding more bytes to the capture. On the other hand there are features, like "Burst Rate" and "Top 3 Values", that seem to be mostly independent from the noise rate. These results are promising, since the fact that some features remain relatively stable even when the noise increases means that detection is still possible with this feature set.

5.2.3 Machine learning algorithms

As mentioned on the previous chapter, after filling the database with the processed captures and their features a choice of which machine learning algorithm to use was made. The *sklearn Python* library has a feature that can estimate the accuracy (ranging from 0 to 1) of an algorithm for a given dataset. Six algorithms were chosen for the preliminary accuracy-prediction tests, with the objective of choosing a single one for the further tests. The results of the preliminary tests are being presented on table 5.3.

Table 5.3: Predicted accuracy scores for the machine learning algorithms

Algorithm	Accuracy	Margin of Error
Logistic Regression	0,189	0,034
Linear Discrimination Analysis	0,617	0,043
K-Neighbors Classifier	0,212	0,054
Decision Tree Classifier	0,605	0,046
Gaussian Naive Bayes	0,486	0,033
Support Vector Machine	0,007	0,008

Two algorithms show greater accuracy than the others: Decision Tree Classifier and Linear Discrimination Analysis. The former was ultimately chosen because even though its projected accuracy slightly lower than that of the former, further testing showed that it had more reliable results. We can therefore conclude that the projected accuracy is meant to be used only as a general indicator, and more testing is always required.

The remaining algorithms showed poor performance, especially in the case of Support Vector Machine. These results show that it is very important to choose an algorithm that is appropriate to the dataset.

5.3 Website detection

After building the dataset, the next step is testing the accuracy of the machine learning algorithm. Using a *Python* script aided by the *sklearn* package, a battery of 500 tests are performed. At this point, the captures with network-level noise are not considered. Each of the tests includes the following steps:

1. Retrieve the data with the features for each of the captures;
2. Split the data in two parts: Train and Validate. The former is used for training the algorithm, while the latter is used to validate the predictions. 15% of the data is used to validate, and the rest for training;
3. Feed the algorithm with the Train data;
4. Perform predictions with the Validate data, using a Decision Tree Classifier;
5. Calculate the confusion matrix and add it to the matrix with the total results.

The fact that this procedure is repeated 500 times confers the results significant statistical meaning, since a large number of combinations of a 85/15 split of the data is tested. In the end, the result is a confusion matrix containing the totals of the predictions for every test. That matrix is represented on 5.5.

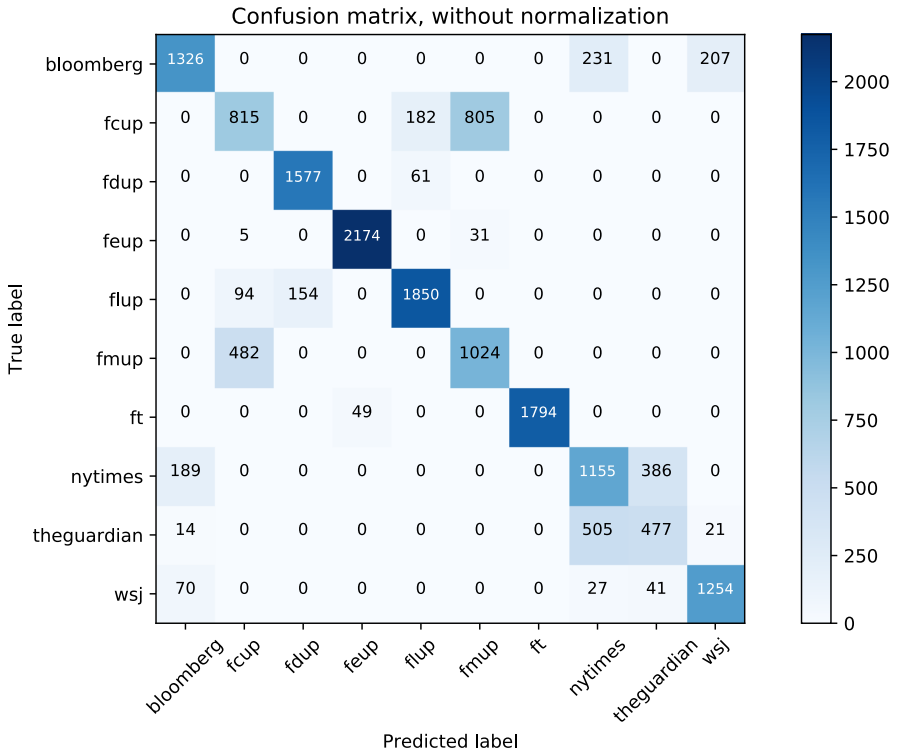


Figure 5.5: Confusion matrix resulting from the battery of tests

Table 5.4: Recall, Precision and F1 scores for each website

Website	Recall	Precision	F1
Bloomberg	0,75	0,83	0,79
FCUP	0,45	0,58	0,51
FDUP	0,96	0,91	0,93
FEUP	0,98	0,98	0,98
FLUP	0,88	0,88	0,88
FMUP	0,68	0,55	0,61
Financial Times	0,97	1,00	0,98
New York Times	0,67	0,60	0,63
The Guardian	0,47	0,53	0,50
Wall Street Journal	0,90	0,85	0,87

Analysis of the confusion matrix shows that some websites are have a nearly perfect detection rate, while others are more easily confused. Websites like FEUP and Financial Times are correctly predicted almost every time, passing the vast majority of the tests. On the other hand, websites like The Guardian and FCUP yield low accuracy rates.

The confusion matrix also lets us see what websites are mixed with each other. For instance, The Guardian and New York Times are frequently identified in an incorrect manner by the algorithm. Another relevant observation is that if we consider the sites by whether they are dynamic or static, we notice that the confusion mostly occurs within the same group, with very few exceptions.

A more precise numerical analysis can be performed by calculating the recall, precision and F1 scores for each website. The recall is given by

$$Precision_i = \frac{Confusion\_Matrix_{ii}}{\sum_j Confusion\_Matrix_{ji}} \tag{5.2}$$

while the recall can be calculated with:

$$Recall_i = \frac{Confusion\_Matrix_{ii}}{\sum_j Confusion\_Matrix_{ij}} \tag{5.3}$$

Finally, the F1 score can be obtained by:

$$F1_i = 2 \times \frac{Precision_i \times Recall_i}{Precision_i + Recall_i} \tag{5.4}$$

Using these formulas, the scores for the websites were calculated. The results, which can potentially range from 0 to 1, are shown on table 5.4, and they confirm the observations that were previously made. We can also see that the F1 score for static and dynamic sites is nearly the same in average, with a value of around 75% for the latter and 78% for the former. Another notable result is that Financial Times sustains a 100% precision score over the 500 tests.

5.4 Influence of network-level noise

So as to evaluate the influence of noise on the algorithm, new tests were performed, this time taking into account the captures with noise. As was previously stated, the captures contemplated five scenarios: No noise and noise levels of 10 KB/s, 50 KB/s, 100 KB/s and 200 KB/s. On all confusion matrices from this point forward, these noise levels are represented in Kbits/s: 80 Kb/s, 400 Kb/s, 800 Kb/s and 1600 Kb/s.

The first tests are the same as in the previous chapter, with the difference that the algorithms is trained with all captures, including those with noise. Likewise, all levels of noise for each website are tested. The new confusion matrix, represented on figure 5.6, and the accuracy scores on table 5.5 show the overall results for every website, without distinguishing the levels of noise.

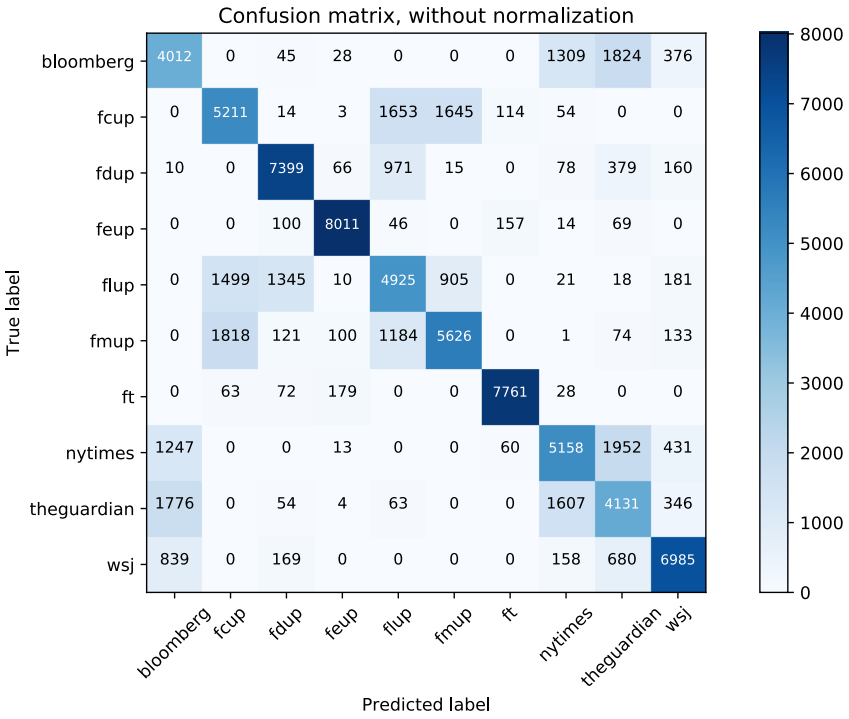


Figure 5.6: Confusion matrix resulting from the battery of tests with noise

A similar pattern to the one observed on figure 5.5 is formed. Once again, the websites that a given website is confused with are, for the most part, of the same class (dynamic or static). However, some sites seem to have lost some of the accuracy, as is the case with Bloomberg and FLUP. In fact, this observation can be confirmed by table 5.5, where we can see that those are the websites with the biggest drop in terms of F1 score, of around 30%. The other websites show smaller, and mostly negative, variations. FCUP and FDUP seem to benefit from the situation, since their F1 score increases, though not too much.

It's worth saying that the number of predictions shown on the confusion matrix on figure 5.6 is larger than those on the matrix on figure 5.6 because the dataset is now five times larger, since for each website captures for each of the four levels of noise are added.

Table 5.5: Recall, Precision and F1 scores for each website (with noise)

Website	Recall	Precision	F1	F1 Δ
Bloomberg	0,53	0,51	0,52	- 0,27
FCUP	0,60	0,61	0,60	+ 0,09
FDUP	0,82	0,79	0,80	- 0,13
FEUP	0,95	0,95	0,95	- 0,03
FLUP	0,55	0,56	0,55	- 0,33
FMUP	0,62	0,69	0,65	+ 0,04
Financial Times	0,96	0,96	0,96	- 0,02
New York Times	0,58	0,61	0,59	- 0,04
The Guardian	0,47	0,45	0,48	- 0,02
Wall Street Journal	0,81	0,85	0,80	- 0,07

In order to make a more in-depth analysis, we need to show how the algorithm reacts to the websites when the network-level noise changes. To that end, individual matrices are generated for each website. We can see two examples in figures 5.7 and 5.8, which correspond to FLUP and Bloomberg respectively. Figure 5.9 contains the information for all the websites, and provides a general view In fact, the individual matrices are only segments of this larger matrix, which is also harder to interpret.

The detection rate of FLUP's website seems to to decrease with the increase of the noise level, with the sharpest drop happening at the 100 KB/s noise level. However, this trend does not continue when the noise is at 200 KB/s, in which the detection rate increases again. The results can be confirmed by observing table 5.6, where we can see that the F1 score steadily decreases as the noise increases, with the exception of 200 KB/s. These results seem to indicate that noise has a direct negative influence on detection, but the irregularity verified on this last case means making such a conclusion is not possible.

On the other hand, Bloomberg's results are more erratic. The F1 score is higher when the noise is at 10 KB/s or 200 KB/s, when compared to no-noise case. However, the other noise rates decrease the accuracy of the algorithm. Therefore we cannot make any meaningful conclusion about the influence of noise in this case. That said, there is a similarity with FLUP: a noise rate of 50 KB/s or 100 KB/s yields the worst results. This is also verified in the case of FEUP and Finantial Times, but it doesn't happen with enough regularity to consider it a trend or to extract conclusions.

Table 5.6: F1 scores for each website and each level of noise

Noise Level	Bloomberg	FCUP	FDUP	FEUP	FLUP	FMUP	Fin. Times	NYT	The Guardian	WSJ
No Noise	0,60	0,64	0,80	0,95	0,86	0,75	0,96	0,42	0,53	0,67
10 KB/s	0,73	0,69	0,73	0,93	0,72	0,83	0,95	0,66	0,56	0,70
50 KB/s	0,42	0,67	0,77	0,92	0,63	0,88	0,87	0,90	0,73	0,95
100 KB/s	0,39	0,91	0,79	0,86	0,54	0,65	0,94	0,54	0,48	0,75
200 KB/s	0,70	0,47	0,86	0,96	0,64	0,84	0,97	0,77	0,58	0,83

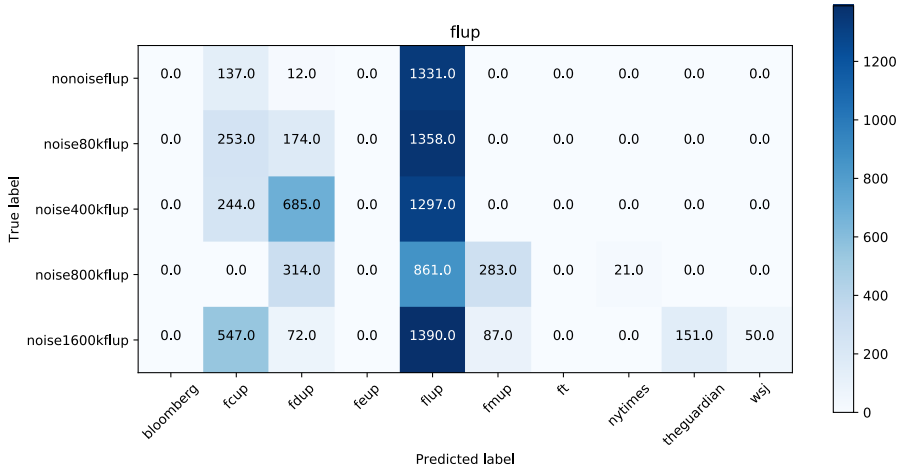


Figure 5.7: Confusion matrix regarding FLUP, with five levels of noise

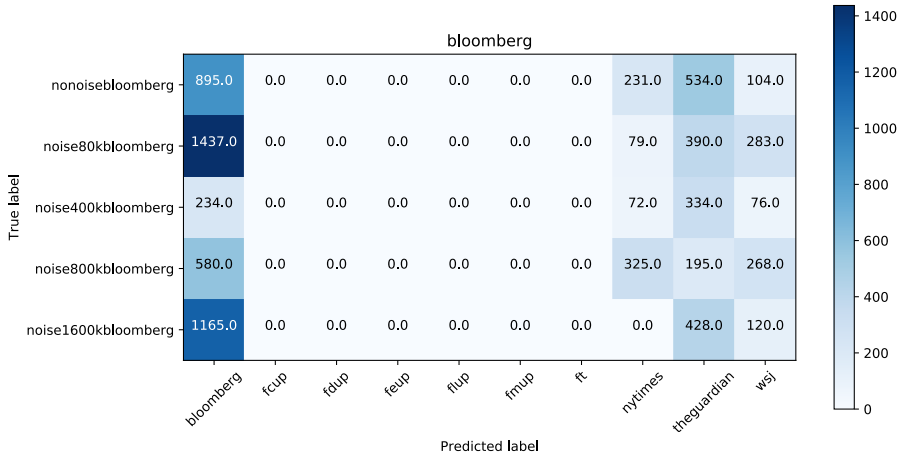


Figure 5.8: Confusion matrix regarding Bloomberg, with five levels of noise

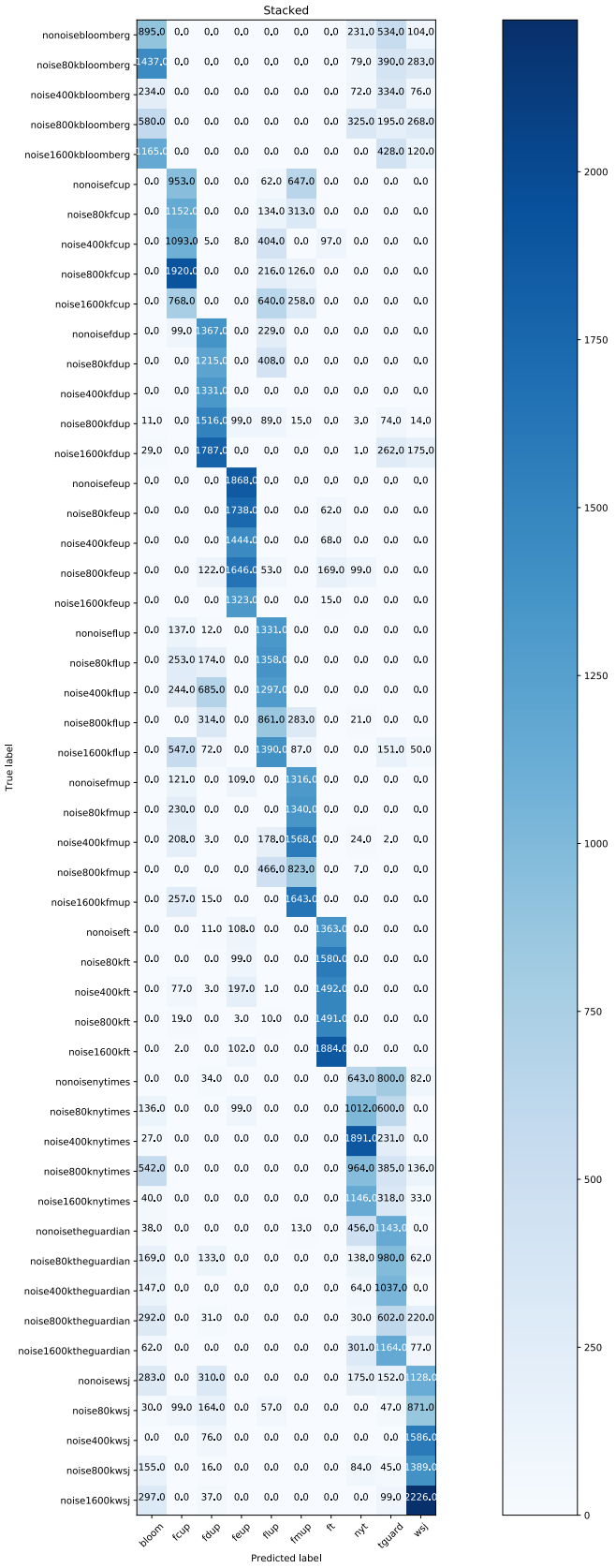


Figure 5.9: Confusion matrix for all websites, with five levels of noise

Chapter 6

Conclusion

The results obtained for this thesis allow for the drawing of some conclusions, which are described in this chapter. A reflection about the results, and how they compare to the expected results, is also included in this chapter, as well as suggestions for future work.

6.1 Analysis of the results

The main objective of this thesis was to devise a model for an attack capable of, based on previously captured data, identify a website. In the case in which no noise is considered, we can say that that objective has been achieved, since all websites had an F1 score of at least 50%, with the overall average being at 76,8%. Two websites, FEUP and Financial Times, came close to a perfect score with 98%. These results are aligned with the expectations, and can be considered positive as they prove an attacker would be able to successfully identify the websites studies in this thesis with a fairly decent success rate.

Conclusions can be made about the importance of how static a website is on the accuracy of the detection process. The average F1 score for both groups - dynamic and static - is very similar, no matter if noise is considered, to the extent that one can affirm that, in the case of the websites tested in this dissertation, this is not an important factor by itself. However, the confusion matrices show that websites tend to be mostly confused with other websites of a similar static level.

The results of the testing with network-level noise yielded mixed results. The expectations were that the application of a constant bitrate UDP traffic during the captures would result in them being more difficult detect with the machine learning algorithm. However, the analysis of tables 5.5 and 5.6 shows that while the F1 score of most sites remains similar (within a difference of 0,1) to the rates when there is no noise, the overall F1 score drops by 0,08 , from 0,77 to 0,69, or 69%.

Furthermore, the original expectations were that for each website, the F1 score would drop as the noise rate increased. That was not observed on the experimental results, and instead the F1 score appeared to vary in an erratic way. In fact, in the case of some websites a certain noise rate had the effect of increasing the F1 score. Therefore, no definite conclusions could be drawn in terms of the influence of network-level noise.

6.2 Future work

There are some improvements that could be made to both the methodology and the solution, since the results show that some questions remains unanswered. Those improvements could be:

- Increase the number of websites included in the testing. This could help determine how the number of websites on the database affects confusion and F1 scores. Within the same website pool, different combinations could be tested.
- Correct the issues with the capture process, especially the *selenium* crashes that compromise some of the captures. They also introduce some delays in the automated capture process, which means that eliminating them would allow for more captures to be done in less time.
- Optimize the processing of the captures. Extracting the data from the captures can be made to be faster, thus reducing significantly the processing time, which can take a lot of time in the current version.
- Devise more features to characterize the captures. In theory, if more features are added and they show variability between websites it is expected that the overall F1 score will improve.
- Test more levels of network-level noise, with an increased bitrate compared to the ones tested in this thesis, so as to be closer to saturating the WiFi interface and affect website detection further. The results obtained with the currently considered noise levels were inconclusive, which can be due to the fact that they are still not high enough to significantly decrease the F1 scores.
- The noise that was generated in the experiments was UDP with a constant bitrate. Other kinds of noise could be tested, such as TCP or SCTP. The traffic can also be sent in a more irregular manner, instead of the current manner in which uniformly sized packets are sent at a nearly constant rate. Sudden bursts of noise traffic can confuse features reliant on traffic shape, as with case of the burst rate.

References

[1] How 802.11 Wireless Works: Wireless. URL: https://technet.microsoft.com/en-us/library/cc757419(v=ws.10).aspx.

[2] J.S. Atkinson, J.E. Mitchell, M. Rio, and G. Matich. Your WiFi is leaking: What do your mobile apps gossip about you? DOI: 10.1016/j.future.2016.05.030, 2016.

[3] Q. Wang, A. Yahyavi, B. Kemme, and W. He. I know what you did on your smartphone: Inferring app usage over encrypted data traffic. pages 433–441, 2015. DOI: 10.1109/CNS.2015.7346855. doi:10.1109/CNS.2015.7346855.

[4] Aaron Smith. U.S. Smartphone Use in 2015, April 2015. URL: http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/.

[5] Internet Statistics & Facts (Including Mobile) for 2016. URL: https://hostingfacts.com/internet-facts-stats-2016/.

[6] Lee Rainie, Sara Kiesler, Ruogu Kang, and Mary Madden. Anonymity, Privacy, and Security Online, September 2013. URL: http://www.pewinternet.org/2013/09/05/anonymity-privacy-and-security-online/.

[7] ICT Facts and Figures 2016. URL: http://www.itu.int:80/en/ITU-D/Statistics/Pages/facts/default.aspx.

[8] IEEE Standard Association - IEEE Get Program. URL: http://standards.ieee.org/getieee802/download/802.11-2012.pdf.

[9] Matthew Gast. *802.11 Wireless Networks: The Definitive Guide*. "O'Reilly Media, Inc.", 2005. Google-Books-ID: 9rHnRzzMHLIC.

[10] Arash Habibi Lashkari, Mir Mohammad Seyed Danesh, and B. Samadi. A survey on wireless security protocols (WEP, WPA and WPA2/802.11i). In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pages 48–52, August 2009. doi:10.1109/ICCSIT.2009.5234856.

[11] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997. Google-Books-ID: EoYBngEACAAJ.

[12] YongBin Zhou and DengGuo Feng. Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing, 2005. zyb@is.iscas.ac.cn 13083 received 27 Oct 2005. URL: http://eprint.iacr.org/2005/388.

[13] Fan Zhang, Wenbo He, Xue Liu, and Patrick G. Bridges. Inferring Users' Online Activities Through Traffic Analysis. In *Proceedings of the Fourth ACM Conference on Wireless Network Security*, WiSec '11, pages 59–70, New York, NY, USA, 2011. ACM. URL: http://doi.acm.org/10.1145/1998412.1998425, doi:10.1145/1998412.1998425.

[14] Tim Stöber, Mario Frank, Jens Schmitt, and Ivan Martinovic. Who Do You Sync You Are?: Smartphone Fingerprinting via Application Behaviour. In *Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '13, pages 7–12, New York, NY, USA, 2013. ACM. URL: http://doi.acm.org/10.1145/2462096.2462099, doi:10.1145/2462096.2462099.

[15] Kehuan Zhang XiaoFeng Wang Shuo Chen, Rui Wang. Side-channel leaks in web applications: a reality today, a challenge tomorrow. In *Proceedings of the IEEE Symposium on Security and Privacy (Oakland)*. IEEE Computer Society, May 2010.

[16] Sharma Sampreet and B.L. Menezes. Implementing side-channel attacks on suggest boxes in web applications. pages 57–62, 2012. DOI: 10.1145/2490428.2490436. doi:10.1145/2490428.2490436.

[17] J.S. Atkinson, M. Rio, J.E. Mitchell, and G. Matich. Your WiFi is leaking: Ignoring encryption, using histograms to remotely detect skype traffic. pages 40–45, 2014. DOI: 10.1109/MILCOM.2014.16. doi:10.1109/MILCOM.2014.16.

[18] M. Li, Y. Meng, J. Liu, H. Zhu, X. Liang, Y. Liu, and N. Ruan. When CSI meets public WiFi: Inferring your mobile phone password via WiFi signals. volume 24-28-October-2016, pages 1068–1079, 2016. DOI: 10.1145/2976749.2978397. doi:10.1145/2976749.2978397.

[19] F. Zhang, W. He, Y. Chen, Z. Li, X. Wang, S. Chen, and X. Liu. Thwarting Wi-Fi side-channel analysis through traffic demultiplexing. *IEEE Transactions on Wireless Communications*, 13(1):86–98, 2014. doi:10.1109/TWC.2013.121013.121473.

[20] Telvis E. Calhoun, Jr., Xiaojun Cao, Yingshu Li, and Raheem Beyah. An 802.11 MAC Layer Covert Channel. *Wirel. Commun. Mob. Comput.*, 12(5):393–405, April 2012. URL: http://dx.doi.org/10.1002/wcm.969, doi:10.1002/wcm.969.

[21] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.*, 11(5):561–580, October 2007. URL: http://dl.acm.org/citation.cfm?id=1367985.1367993.